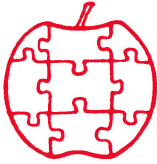


Apple

\$1.80



Assembly

Line

Volume 5 -- Issue 7

April, 1985

In This Issue...

Putting S-C Macro on a quikLoader Card	2
New Book: Inside the Apple //c.	7
Volume Catalog for Corvus and Sider.	9
Shrinking Code Inside ProDOS	12
Fast Text Windows for Applesoft.	16
8086/8088 Cross Assembler.	21
Powerful 65816 Board on the Horizon.	22
USR Command to List Major Labels Only.	24
Review of the FCP Sider Hard Disk.	27

A New Book Appears

Jim Sather's new book, *Understanding the Apple //e*, arrived today. We'll have a complete review next month, but at first glance it looks even better than his first book. Check our ad on page 3 for pricing.

And an Old Book Reappears

Roger Wagner Publishing has obtained the rights to Roger's "Assembly Lines -- the Book" from Softalk. A new edition is now available, still at \$19.95. We sold hundreds of copies of this book, which in excellent tutorial fashion leads a beginner into the fascinating world of assembly language. "Assembly Lines -- the Disk" is also available, with all the sample source code formatted for the Merlin assembler. If you wish to order the book from us, our price is only \$18 plus shipping.

Postage Increases

The recent Post Office rate increases had little effect on the Bulk and First Class rates, only \$.015-.03 per piece, or \$.18-.36 per year per subscription. We'll accept that much of a cost increase. Foreign Air Mail is another matter, though. Those rates went up by \$.16-.19 per piece, or \$1.92-2.28 per year per subscription. Therefore, the foreign subscription rate is now \$32 per year.

Putting S-C Macro on a QuikLoader Card.....Jan Eugenides

The QuikLoader by Southern California Research Group is one of those rare devices that causes you to wonder how you ever got along without one. I have had mine for about a year now, and I would never go back to the old way of loading programs!

Briefly, the QuikLoader allows you to put whatever programs you desire on EPROMS, which then plug into the QuikLoader. EPROMS from 2716-27512 can be used, for a possible 512K bytes of program space on one QuikLoader (equivalent to four Apple floppies!). You can have more than one card, of course, so there's lots of room available for just about anything. The QuikLoader also comes with DOS 3.3 already installed, along with FID, and COPYA. When you turn on your machine, you'll hear a little whoop instead of the familiar beep. DOS has just been loaded in about 2 seconds. No more booting! In fact, I seldom put DOS on a disk anymore, and I can use the space for programs instead.

Programs which are on the QuikLoader can be loaded into RAM and executed in about 2 seconds, with just two keystrokes! Since they are loaded into their regular RAM locations, they do NOT need to be modified in any way.

You can see a catalog of the QuikLoader by typing "Q" followed by RESET. The program names appear with letters A-Z next to them. Then you can select and run the programs by typing the letter corresponding to that program. Alternatively, if you want to run the primary routine on a chip, just press the number of the socket it is in followed by RESET. More on this later.

Putting programs on the QuikLoader is somewhat problematical, however. The manual is STILL in it's draft form, although they have been promising a better one for over a year. Oh well...a little trial and error is good for the soul.

In order to put the S-C Macro Assembler on the QuikLoader, it is necessary to write what's known as a "primary" routine. The QuikLoader has a built-in operating system which allows you to move blocks of memory to their RAM locations from the various EPROMS on the QuikLoader card, and then execute them however you wish. The following program is intended to be used on a 27128 EPROM, which will hold the entire S-C Macro Assembler, with driver (I used the Ultraterm driver for this program) and the Fast Bload patches, which I chose to load between DOS and its buffers, rather than actually patch the DOS. You can do it either way, it's up to you.

This program is called the "overhead" for the EPROM. It goes at \$FEB0 in the actual chip. The catalog must appear at \$FF00. These are the addresses as the Apple would see them, not the absolute addresses relative to the chip. A 27128 will address as though it runs from \$C000 to \$FFFF as far as the Apple is concerned. In other words, the chip's address \$0000 equals the Apple's address \$C000. Things are further complicated by the fact that an Apple II+ cannot address the range from \$C000 to

S-C Macro Assembler Version 2.0.....\$100
 Version 2.0 Upgrade Kit for 1.0/1.1/1.2 owners.....\$20
 Source Code for Version 1.1 (on two disk sides).....\$100
 Full Screen Editor for S-C Macro (with complete source code).....\$49
 S-C Cross Reference Utility (without source code).....\$20
 S-C Cross Reference Utility (with complete source code).....\$50
 DISASM Dis-Assembler (RAK-Ware).....\$30
 Source Code for DISASM.....additional \$30
 S-C Word Processor (with complete source code).....\$50
 DP18 Source and Object.....\$50
 Double Precision Floating Point for Applesoft (with source code).....\$50
 S-C Documentor (complete commented source code of Applesoft ROMs).....\$50
 Source Code of //e CX & P8 ROMs on disk.....\$15

(All source code is formatted for S-C Macro Assembler. Other assemblers require some effort to convert file type and edit directives.)

AAL Quarterly Disks.....each \$15, or any four for \$45

	1980	1981	1982	1983	1984	1985
Each disk contains	Jan-Mar	Apr-Jun	Jul-Sep	Oct-Dec		
the source code from	2	3	4	5		
three issues of AAL,	6	7	8	9		
saving you lots of	10	11	12	13		
typing and testing.	14	15	16	17		
	18					

AWIIE Toolkit (Don Lancaster. Synergetics).....\$39
 Visible Computer: 6502 (Software Masters).....(reg. \$50) \$45
 ES-CAPE: Extended S-C Applesoft Program Editor (new price. was \$60) \$40
 "Bag of Tricks", Worth & Lechner, with diskette.....(\$39.95) \$36

Blank Diskettes (Verbatim).....package of 20 for \$32
 (Premium quality, single-sided, double density, with hub rings)

Vinyl disk pages, 6"x8.5", hold two disks each10 for \$6

Diskette Mailing Protectors (hold 1 or 2 disks).....40 cents each

(Cardboard folders designed to fit 6"x9" Envelopes.) or \$25 per 100

Envelopes for Diskette Mailers.....6 cents each

quikLoader EPROM System (SCRG).....(\$179) \$170
 PROMGRAMMER (SCRG).....(\$149.50) \$140
 D Manual Controller (SCRG).....(\$90) \$85
 Switch-a-Slot (SCRG).....(\$190) \$175
 Extend-a-Slot (SCRG)(\$35) \$32
 Write Guard Disk Mod Kit (Mark IV).....\$45

Books. Books, Books.....compare our discount prices!

"Inside the Apple //c", Little(\$19.95) \$18
 "Inside the Apple //e", Little(\$19.95) \$18
 "Apple II+/Iie Troubleshooting & Repair Guide", Brenner (\$19.95) \$18
 "Apple II Circuit Description", Gayler (\$22.95) \$21
 "Understanding the Apple II", Sather (\$22.95) \$21
 "Understanding the Apple //e", Sather (\$24.95) \$23
 "Enhancing Your Apple II, vol. 1", Lancaster (\$15.95) \$15
 "Assembly Cookbook for the Apple II/Iie", Lancaster (\$21.95) \$20
 "Incredible Secret Money Machine", Lancaster (\$7.95) \$7
 "Beneath Apple DOS", Worth & Lechner (\$19.95) \$18
 "Beneath Apple ProDOS", Worth & Lechner (\$19.95) \$18
 "What's Where in the Apple", Second Edition..... (\$19.95) \$19
 "6502 Assembly Language Programming", Leventhal..... (\$18.95) \$18
 "6502 Subroutines", Leventhal..... (\$18.95) \$18
 "Real Time Programming -- Neglected Topics", Foster..... (\$9.95) \$9
 "Microcomputer Graphics", Myers..... (\$12.95) \$12
 "Assem. Language for Applesoft Programmers", Finley & Myers. (\$16.95) \$16
 "Assembly Lines -- the Book", Wagner..... (\$19.95) \$18

Add \$1 50 per book for US shipping. Foreign orders add postage needed.

Texas residents please add 6 1/8 % sales tax to all orders.

*** S-C SOFTWARE. P. O. BOX 280300, Dallas, TX 75228 ***
 *** (214) 324-2050 ***
 *** We accept Master Card, VISA and American Express ***

\$C7FF without a small circuit modification. In this case it's no problem, the space from \$C800-\$FFFF is more than enough to house the entire assembler. If you needed more space, you could put your primary routine in the \$C000-\$C7FF space.

The rest of the EPROM contains the code for the assembler itself, and the fast Bload patch. The assembler goes from \$C800-\$EFFF, and the Bload patch from \$F000 to \$F04D. You must pack these files together in RAM somewhere prior to burning the chip. In other words, Bload the assembler at, say, \$2800-4FFF. Put the Bload patch at \$5000-504D. Then Bload the overhead program at \$5EB0. The rest of the EPROM doesn't matter. Then burn all this stuff into the EPROM starting at \$800 relative to the chip. Thus, when you install the chip on the card, it will show up at \$C800-FFFF like it should. If your EPROM burner won't burn partial chips, just start the burn from \$2000 and it'll work out.

That's it. Just install the chip on the QuikLoader in any socket. To run the assembler just type the socket number followed by RESET. In two seconds the assembler will load and start! No more waiting to boot DOS, load the program, etc. You don't even have to look for a disk! Sure speeds up the work.

This should help augment the information in the manual a little, and get you on your way. I have installed the S-C assembler, Rak-ware's DISASM, a modified SOURCEROR (it now outputs S-C format code, heh heh), the S-C Word Processor, a terminal program of my own design (it's capture buffer exactly coincides with the S-C Word Processor buffer! I can come off-line and begin editing with two keystrokes, and no disk access!), and some other utilities. All stored inside the Apple, available instantly at any time. For \$170 (the price from S-C Software), the QuikLoader is a MUST.

By the way, for a reasonable fee I will install programs on EPROMS for you. You supply the programs and EPROMS, and I'll do the rest. Some programs are not suitable...particularly those which access the disk a lot. They would require extensive modification and that's best left to the original author. Also, copy-protected stuff cannot be loaded, because there's no way to go at the files. Contact me if you're interested, at 11601 NW 18th St., Pembroke Pines, FL 33026.

[For \$20, S-C Software will send registered owners of version 2.0 a 27128 with the S-C Macro Assembler on it. This adds five lines to the QuikLoader menu, allowing you to choose the screen driver you wish. Only the \$D000 (language card) version is provided.]

Here's the overhead program, with GETSLOT overhead taken from the QuikLoader manual.

```

1000 *SAVES-ASSEM.2.0.OH(FAST BLOAD)
1010 *-----
1020 *1/31/85
1030 *-----
1040 *
1050 *S-C MACRO ASSEMBLER OVERHEAD - ULTRATERM VERSION
1060 *      by Jan Eugenides
1070 * 3/9/85
1080 *
1090 *-----
1100 *CHIP 0 ROUTINE EQUATES
1110 *-----
1120 *Y-register indexes of the chip 0 routines
1130 *-----
1140 *
00- 1150 MOVEBLK .EQ 0      Move data block to RAM
08- 1160 GOMBRD .EQ 8      Go to motherboard
1170 *-----
1180 *
1190 * GENERAL EQUATES
1200 *-----
1210 *
26- 1220 PRISLOT .EQ $26   Storage for primary slot
2D- 1230 QLMAP .EQ $2D    bitmap of QL slots
3A- 1240 SRCL .EQ $3A    indirect source
020A- 1250 SAVCTRL .EQ $20A save control word
C081- 1260 QLCTRL .EQ $C081 QL control register
1270 *-----
1280 *
1290 * GET SLOT EQUATES
1300 *-----
1310 *
18- 1320 QLOFF .EQ $18     00011000 QLOFF; CHIP 0
20- 1330 CHKNUM .EQ $20    NUMBER OF FIND SLOT CHECKS
40- 1340 GSCL .EQ $40     GET SLOT C PARAMETER.
41- 1350 GSCH .EQ $41
42- 1360 GSEL .EQ $42     GET SLOT E PARM
43- 1370 GSEH .EQ $43
C006- 1380 SLTXROM .EQ $C006 IIE SOFT SWITCH
C00A- 1390 INT3ROM .EQ $C00A "
C00B- 1400 SLT3ROM .EQ $C00B "
CFFF- 1410 CLRRROM .EQ $CFFF
1420 *-----
1430 .OR $FEB0
1440 .TF ASM.2.0.OH
1450 *-----
1460 * This program will start the assembler in 80x32
1470 * mode with ultraterm. Assumes that assembler has
1480 * been patched at $DBC9 and $DC11 for 32 line mode.
1490 * i.e. the normal $17 is now $1F. If mode is changed
1500 * these bytes must be re-patched. ($2F for 48 line mode)
1510 * For S-C assembler 2.0 March 1985 version with Bob's
1520 * ultraterm driver attached at $F700.
1530 *-----
FEB0- A9 00 1540 START.PROG LDA #0      Turn on Ultraterm
FEB2- 20 00 C3 1550 JSR $C300
FEB5- A9 16 1560 LDA #22      bring up in 80x32 mode
FEB7- 20 ED FD 1570 JSR $FDED
FEBA- A9 B5 1580 LDA #5       Mode 5
FEBB- 20 ED FD 1590 JSR $FDED
FEBF- A9 CB 1600 LDA #$CB
FEC1- 8D D1 03 1610 STA $3D1     set warmstart vector
FEC4- A9 00 1620 LDA #0
FEC6- 8D 00 9D 1630 STA $9D00    make room between DOS a buffers
FEC9- 20 D4 A7 1640 JSR $A7D4    for fast BLOAD patch
FECC- A9 30 1650 LDA #$30
FECE- 8D A6 AC 1660 STA $ACA6    patch dos to call fast Bload
FED1- A9 9C 1670 LDA #$9C
FED3- 8D A7 AC 1680 STA $ACA7    which is now at $9C30
FED6- A9 4C 1690 LDA #$4C
FED8- 8D 00 E0 1700 STA $E000
FEDB- A9 00 1710 LDA #0
FEDD- 8D 01 E0 1720 STA $E001    put assembler coldstart vector at $E000
FEE0- A9 D0 1730 LDA #$D0
FEE2- 8D 02 E0 1740 STA $E002
FEE5- AD 80 C0 1750 LDA $C080    select ram card
FEE8- 4C 00 D0 1760 JMP $D000    coldstart assembler
1770 SP.END

```

```

1780 *-----
1790 .BS $FF00-* SKIP TO FF00
1800 *-----
1810 *KATALOG ENTRIES START HERE
1820 *-----
FF00- 90 1830 ASMK .DA #$90 PRIMARY
FF01- 9F FF 1840 .DA N.RESET SOURCE
FF03- 00 00 1850 .DA $0000 LENGTH
FF05- 00 00 1860 .DA $0000 DESTINATION
FF07- C1 D3 CD 1870 .AS -"ASM"
1880 *-----
FF0A- 86 1890 .DA #$86 END OF KAT RECORD
1900 *-----
FF0B- 00 C8 1910 ASMPARM1 .DA $C800 SOURCE assembler + driver goes here
FF0D- FF 27 1920 .DA $27FF LENGTH will load from $D000-$F7FF
FF0F- 00 D0 1930 .DA $D000 DESTINATION
FF11- 00 F0 1940 ASMPARM2 .DA $F000 SOURCE fast blood routine
FF13- 4D 00 1950 .DA $004D LENGTH
FF15- 30 9C 1960 .DA $9C3D
1970 *-----
FF17- 4A 1980 INVERT LSR
FF18- 6A 1990 ROR
FF19- 6A 2000 ROR
FF1A- 6A 2010 ROR
FF1B- 29 E0 2020 AND #$E0
FF1D- 8D 0A 02 2030 STA SAVCTRL
FF20- 60 2040 RTS
2050 *-----
FF21- 2060 .BS $FF53-* SKIP TO FF53
FF53- A9 18 2070 OFFFLP LDA #QLOFF
FF55- 9D 81 CO 2080 STA QLCTRL,X TURN OFF THE QL
FF58- 20 5D FF 2090 RTSLOC JSR GETSLOT THIS INSTRUCTION AT $FF58 (RTS)
FF5B- D0 F6 2100 BNE OFFFLP
2110 *-----
2120 *
2130 * FIND SLOT NUMBER BY COMPARING CNXX TO ENXX FOR EACH SLOT
2140 * START WITH SLOT 7 USR MUST BE RESET FOR SEARCH TO BE
2150 * EFFECTIVE IN II OR IIE.
2160 *-----
FF5D- 8D 06 CO 2170 GETSLOT STA SLTXROM ENABLE IIE I/O SELECTS
FF60- 8D 0B CO 2180 STA SLT3ROM
FF63- A9 00 2190 LDA #0
FF65- 85 40 2200 STA GSCL
FF67- 85 42 2210 STA GSEL
FF69- A9 C1 2220 TRYAGEN LDA #$C1 START WITH SLOT 1
FF6B- 85 41 2230 STA GSCH
FF6D- A9 E1 2240 LDA #$E1
FF6F- 85 43 2250 STA GSEH DESTINATION = $EN00
FF71- A0 20 2260 LDY #CHKNUM GET NUMBER OF CHECKS TO VERIFY
FF73- B1 40 2270 LOOKLP LDA (GSCL),Y
FF75- D1 42 2280 CMP (GSEL),Y
FF77- D0 19 2290 BNE NOTHERE BRANCH IF QL NOT IN THIS SLOT
FF79- 88 2300 DEY
FF7A- D0 F7 2310 BNE LOOKLP
FF7C- A5 41 2320 LDA GSCH
FF7E- A8 2330 TAY
FF7F- 0A 2340 ASL GET SLOTNUM TIMES $10 TO X
FF80- 0A 2350 ASL
FF81- 0A 2360 ASL
FF82- 0A 2370 ASL
FF83- AA 2380 TAX
FF84- B9 86 FE 2390 LDA $FE86,Y GET BIT MAP
FF87- 05 2D 2400 ORA QLMAP
FF89- 85 2D 2410 STA QLMAP SET BIT IN QLMAP
FF8B- 8D 0A CO 2420 STA INT3ROM LEAVE INT3ROM AS NORMAL RESET DOES
2430 *-----
2440 *NORMAL RESET FORCES SLTXROM
2450 *LEAVE 3ROM AND XROM AS WITH NORMAL RESET
2460 *-----
FF8E- AD FF CF 2470 LDA CLRROM EXPANSION ROM OFF
FF91- 60 2480 RTS
FF92- E6 41 2490 NOTHERE INC GSCH
FF94- E6 43 2500 INC GSEH CHECK IN NEXT SLOT
FF96- D0 DB 2510 BNE LOOKLP BRANCH ALWAYS
2520 *-----
2530 *EQU $CO SHOULDN'T OCCUR; BOMB IF DOES
2540 *-----

```

```

FF98- 80      2550 MAP      .DA #$80
FF99- 40      2560      .DA #$40
FF9A- 20      2570      .DA #$20
FF9B- 10      2580      .DA #$10
FF9C- 08      2590      .DA #$08
FF9D- 04      2600      .DA #$04
FF9E- 02      2610      .DA #$02
2620 *-----*
2630 * THIS IS N.RESET ROUTINE OF THIS CHIP
2640 *-----*
FF9F- 20 17 FF 2650 N.RESET      JSR INVERT      Invert the control word
FFA2- A0 05 2660      LDY #5
FFA4- B9 0B FF 2670 .1      LDA ASMPARM1.Y      Move ASSEMBLER parms
FFA7- 99 3A 00 2680      STA SRCL Y
FFAA- 88      2690      DEY
FFAB- 10 F7      2700      BPL .1
FFAD- AD 0A 02 2710      LDA SAVCTRL      get control word
FFB0- A6 26      2720      LDX PRISLOT      Slot in X reg
FFB2- A0 00      2730      LDY #MOVEBLK      Command index for move block routine
FFB4- 20 EC FF 2740      JSR GOCHIP0      Call chip 0 to move block
FFB7- A0 05      2750      LDY #5
FFB9- B9 11 FF 2760 .2      LDA ASMPARM2.Y      Move Fast Bload routine parms
FFBC- 99 3A 00 2770      STA SRCL,Y
FFBF- 88      2780      DEY
FFC0- 10 F7      2790      BPL .2
FFC2- AD 0A 02 2800      LDA SAVCTRL      get control word
FFC5- A6 26      2810      LDX PRISLOT      Slot in X reg
FFC7- A0 00      2820      LDY #MOVEBLK      Command index - move block
FFC9- 20 EC FF 2830      JSR GOCHIP0      Call chip 0
FFCC- A0 3B      2840      LDY #SP.END-START.PROG
FFCE- B9 80 FE 2850 .3      LDA START.PROG,Y Move startup program to $300
FFD1- 99 00 03 2860      STA $300,Y
FFD4- 88      2870      DEY
FFD5- 10 F7      2880      BPL .3
FFD7- A9 02      2890      LDA #$02      put address-1 on stack
FFD9- 48      2900      PHA
FFDA- A9 FF      2910      LDA #$FF
FFDC- 48      2920      PHA
FFDD- A0 08      2930      LDY #GOMRBRD      jmp to $300 to start
FFDF- AD 0A 02 2940      LDA SAVCTRL
FFE2- A6 26      2950      LDX PRISLOT
FFE4- 4C EC FF 2960      JMP GOCHIP0
2970 *-----*
FFE7-      2980      .BS $FFEC-*      SKIP TO FFEC
FFEC- 9D 81 C0 2990 GOCHIP0      STA OLCTRL,X      GO TO CHIP 0
FFEF- 4C 9F FF 3000      JMP N.RESET      DO N.RESET ROUTINE OF THIS CHIP
FFF2-      3010      .BS 3
FFF5- 60      3020      RTS
FFF6-      3030      .BS 2
FFF8- 00 FF      3040      .DA ASMK      FIRST KATALOG LOCATION
FFFA- FB 03      3050      .DA $3FB      NMI VECTOR

```

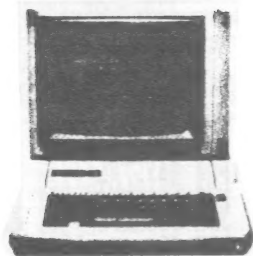
New Book: Inside the Apple //c

What Gary Little did for the //e he has repeated for the //c. Of course a lot of the material is the same for both computers and both books, but there is much new material. If you have a //c and not a //e, then this book will be much more helpful.

For one thing, when explaining assembly language he includes the new opcodes and address modes of the 65C02. For another, the chapter on Disk Operating Systems is now 100% ProDOS, and includes more detail on ProDOS than the //e book. Naturally, since the //c has no cassette port or I/O slots, that material has been left out. On the other hand there is a lot of new data about the Apple mouse port and the built-in serial ports.

The book is published by Brady (Prentice-Hall), is 363 + xv pages, and sells for \$19.95. (We'll send you one for a little less, see page 3 of this newsletter)

The Number One, Best Selling, Expandable Ram Card for the Apple Is RAMWORKS™



A Few Reasons Why!

It's Appleworks' Best Friend

A 64K RAMWORKS will add 46K to your available desktop memory, a 128K RAMWORKS will add 91K, a 256K RAMWORKS will add 182K, and 512K RAMWORKS will add 364K and a 1 meg RAMWORKS will give you nearly an 800K DESKTOP. But RAMWORKS does more than just increase the desktop memory (as if that weren't enough). With RAMWORKS, Appleworks will be able to run up to 20 times faster. And now only RAMWORKS does this all automatically with no need for any optional software. Appleworks is a very large program. So large, in fact, that it won't all fit in a 128K Apple, so Appleworks has to constantly go back to the disk drive for more data. When you have a RAMWORKS card, you can have enough memory so that the entire Appleworks program can be loaded into the computer. This greatly increases the speed at which Appleworks operates because the disk drive operates at 300 RPM. RAMWORKS operates at the speed of light!

It's Compatible

The simple fact is that RAMWORKS is compatible with more off-the-shelf software than any other RAM card. RAMWORKS is 100% compatible with ALL software written for the Apple II column and extended 80 column card. Additionally, RAMWORKS can emulate other RAM cards so software written for other cards will run without modification. Software written for RAMWORKS will not work on other cards. We can emulate others, but others can't emulate us.

It's Easy

RAMWORKS just plugs in and clear instructions show how to plug more memory chips into RAMWORKS as your needs increase.

It's Popular

Popularity translates into great software support because software companies can't support all RAM cards, they can only support the ones their customers are likely to own.

It's Reasonable

Applied Engineering's years of experience and wide product line really pays off, and because of our high sales levels we buy most of our I.C. chips factory direct. So don't let our low prices fool you, they're caused by high volume production. That's why we can offer the most memory for the least money.

It's In Color

For only \$129 an RGB option can be added to RAMWORKS to give you double high resolution color graphics and sharp 80 column text on any Apple compatible RGB monitor. The RGB option does not take up another slot, but rather it plugs into RAMWORKS. You can order the RGB option with your RAMWORKS or add it on at a later date.

It Corrects Mistakes

Let's say you bought some other RAM card (and that's a mistake) and your RAM card is not being recognized by Appleworks, Advanced VisiCalc, FlashCalc or other programs and you want RAMWORKS. No problem. The memory chips on the card that you have now, which is where most of the money is, can be unplugged and then plugged into the expansion sockets on RAMWORKS.

It's Got It All

- ☒ 80 Column Text
- ☒ Double high resolution graphics (with or without RGB option)
- ☒ User Expandable to 1 Megabyte
- ☒ Can Use 64K or 256K RAMS in any combination
- ☒ Adds Memory to Appleworks
- ☒ Accelerates Appleworks
- ☒ 100% Compatible with All IIe software
- ☒ RAM Disk software available, compatible with PRO-DOS, DOS 3.3, and PASCAL (\$29.00)
- ☒ RAM Disk available for CP/M (\$29.00)
- ☒ VisiCalc preboot available (\$29.00)
- ☒ RGB option
- ☒ Takes only one slot
- ☒ 3 year no hassle warranty

Ramworks™ with 64K Installed . . .	\$179
Ramworks™ with 128K Installed . . .	\$249
Ramworks™ with 256K Installed . . .	\$399
Ramworks™ with 512K Installed . . .	\$649
Ramworks™ with 1 MEG Installed	\$1199
RGB Option (can be added later) . . .	\$129

Call (214) 492-2027

9 a.m. to 11 p.m. 7 days a week or

Send check or money order to:

P. O. Box 798

Carrollton, Texas 75006

MasterCard, Visa and C.O.D. welcome.

No extra charge for credit cards.

Texas residents add 5% sales tax. Add \$10.00 if outside U.S.A.

When you want to expand your IIe's memory and make Appleworks and other programs have more memory and go faster, why not do what Apple computer does, call Applied Engineering and ask for RAMWORKS.

AE
APPLIED ENGINEERING

Volume Catalog for Corvus and Sider.....Bob Sander-Cederlof

When I have a stack of floppies, I can quickly shuffle through them reading labels to find the two or three most likely to have the elusive file I want. On a hard disk it is hard to read the labels....

The last time I had a Corvus sitting in this room, there was a program on the utility disk which would list the first file name from each volume. If you were careful about making the first file name descriptive, it could act like a label. Of course, nearly every floppy around here has a first file named HELLO. Not too helpful.

Several years ago Bill Morgan wrote a program we published in AAL called the Catalog Arranger. It allows you to re-arrange the filenames in any catalog to any order you wish, and to rename the files using any combination of upper/lower case, inverse, flashing, and control-characters. I use Catalog Arranger to make a "title" file at the beginning of each hard disk volume. (If you never heard of Catalog Arranger, you can type it in from AALs of October 1982 and January 1983. It is also available on a Quarterly disk for only \$15.)

Now that I don't have the Corvus, or its handy program for listing the names of the first file in each volume, I decided to write my own. The program that follows prints out the volume number, two spaces, and then the name of the first file. If the volume is empty, it prints "<<<EMPTY VOLUME>>>". You can abort the listing by pressing RETURN or ESCAPE, or pause it by pressing any other key.

Lines 1090-1100 set the origin at \$803 and cause the object program to be written on a BRUNnable file called CAT. We write it at \$803 rather than \$800 so that Applesoft will work correctly after CAT is finished. Applesoft gets upset if \$800 has any non-zero value in it.

I used two monitor routines. \$FD8E prints a carriage return, and \$FDED prints any character from the A-register.

I also used routines inside DOS. \$AFF7 reads the VTOC of the current volume, using the inverse volume number from the variable R.VOLUME. If there is any error in trying to read the VTOC, DOS would normally go through its procedure of printing the message and returning to Applesoft. We cannot allow that, so I install a temporary patch to make the error condition cause a return to my code with carry set. If there is no error, carry will be clear. The only likely error is that I am asking for the VTOC of a non-existing volume, which means I have already processed them all. The patching, call, and de-patching take place in lines 1160-1220. Line 1230 branches to my exit routine if there was an error reported.

I also call on \$B011 to read the first sector of the catalog. If you call \$B011 with carry clear it reads the first sector of the catalog; with carry set, it reads the next sector of the catalog. The sector is read into a standard buffer at

\$B4BB-\$B5BA. See "Beneath Apple DOS" for a complete description of the catalog sectors.

Lines 1270-1440 convert the volume number to decimal and print it out. Lines 1450-1480 check for an empty directory. If it is empty, lines 1740-1800 print the empty volume message. Otherwise, lines 1490-1550 print the file name. Right here my program could use some improvement. It is possible for an empty volume to not look empty, because deleted files are not physically removed from the catalog. The byte we check for an empty volume could have \$FF in it, signifying a deleted file. In this case my program should continue searching through the catalog for either the end or a non-deleted file. I didn't think it was absolutely necessary, since I was using Catalog Arranger to remove all deleted files from the catalog and position the title line at the very top.

Line 1730 returns back to DOS by JMP \$3D0. This reminds me of glitch we all run into from time to time. If you intend to BRUN a program from the command level of the assembler or of Applesoft, it needs to end with JMP \$3D0. Ending with an RTS will not do, because BRUN does not leave any return address on the stack. On the other hand, if you intend to start the program by using a CALL or MGO or \$...G command, it is all right to end with an RTS. In fact, with a CALL from inside a running Applesoft program you MUST use an RTS. Just something to watch out for.

```

1000 *SAVE S.HARD CAT
1010 *-----
03D9- 1020 RWT$      .EQ $03D9
03E3- 1030 GETIOB   .EQ $03E3
1040 *-----
B4BB- 1050 CATALOG.BUFFER .EQ $B4BB
1060 *-----
B5F9- 1070 R.VOLUME   .EQ $B5F9
1080 *-----
1090      .OR $803
1100      .TF CAT
1110 *-----
1120 HARD.CAT
0803- 20 8E FD 1130      JSR $FD8E
0806- A9 FE 1140      LDA #$FE      FOR VOLUME=1 TO 254
0808- 8D F9 B5 1150      STA R.VOLUME   (.EOR.FF OF VOLUME #)
1160 *---PATCH DOS TO TRAP ERROR---
080B- A9 60 1170      .1 LDA #$60      'RTS'
080D- 8D 9E B0 1180      STA $B09E
0810- 20 F7 AF 1190      JSR $AFF7      READ VTOC OF VOLUME
1200 *---REMOVE PATCH---
0813- A9 B0 1210      LDA #$B0      'BCS'
0815- 8D 9E B0 1220      STA $B09E
0818- B0 5A 1230      BCS .7      OUT OF LOOP, BEYOND LAST VOLUME
1240 *---READ 1ST CATALOG SECTOR---
081A- 18 1250      CLC
081B- 20 11 B0 1260      JSR $B011
1270 *---PRINT VOLUME #---
081E- AD F9 B5 1280      LDA R.VOLUME   INVERSE OF #
0821- 49 FF 1290      EOR $FF      BACK TO NORMAL FORM
0823- A2 B0 1300      LDX #0"      CONVERT TO DECIMAL
0825- C9 0A 1310      .2 CMP #10      ANY 10'S?
0827- 90 05 1320      BCC .3      ...NONE LEFT
0829- E9 0A 1330      SBC #10      ...YES, DIMINISH
082B- E8 1340      INX      AND COUNT IT
082C- D0 F7 1350      BNE .2      ...ALWAYS
082E- 48 1360      PHA      SAVE UNITS
082F- 8A 1370      TXA      PRINT TENS
0830- 20 ED FD 1380      JSR $FDED
0833- 68 1390      PLA      GET UNITS
0834- 09 B0 1400      ORA #0"      AND PRINT IT
0836- 20 ED FD 1410      JSR $FDED

```

```

0839- A9 A0 1420 LDA # " PRINT " "
083B- 20 ED FD 1430 JSR $FDED
083E- 20 ED FD 1440 JSR $FDED
1450 *---PRINT NAME OF FIRST FILE-----
0841- A0 0B 1460 LDY #11
0843- B9 BB B4 1470 LDA $B4BB,Y
0846- F0 32 1480 BEQ .8 ...EMPTY VOLUME
0848- A2 00 1490 LDX #0
084A- B9 BE B4 1500 .4 LDA $B4BB+3,Y
084D- C8 1510 INY
084E- 20 ED FD 1520 JSR $FDED
0851- E8 1530 INX
0852- E0 1E 1540 CPX #30
0854- 90 F4 1550 BCC .4
1560 *---PRINT CARRIAGE RETURN-----
0856- 20 8E FD 1570 .5 JSR $FD8E
1580 *---NEXT VOLUME-----
0859- CE F9 B5 1590 DEC R.VOLUME
1600 *---POSSIBLE PAUSE OR ABORT-----
085C- AD 00 C0 1610 LDA $C000 ANY KEY PAUSES
085F- 10 AA 1620 BPL .1 NO KEY
0861- 8D 10 C0 1630 STA $C010
0864- C9 8D 1640 CMP #$8D <RETURN> ABORTS
0866- F0 0C 1650 BEQ .7
0868- AD 00 C0 1660 .6 LDA $C000 PAUSE LOOP
086B- 10 FB 1670 BPL .6
086D- 8D 10 C0 1680 STA $C010
0870- C9 8D 1690 CMP #$8D AGAIN. RETURN AGORTS
0872- D0 97 1700 BNE .1
1710 *-----
0874- 20 8E FD 1720 .7 JSR $FD8E <RETURN>
0877- 4C D0 03 1730 JMP $3D0 BACK TO DOS
1740 *---EMPTY VOLUME-----
087A- A2 00 1750 .8 LDX #0
087C- BD 87 08 1760 .9 LDA MT.X PRINT STRING BELOW
087F- F0 D5 1770 BEQ .5
0881- 20 ED FD 1780 JSR $FDED
0884- E8 1790 INX
0885- D0 F5 1800 BNE .9 ...ALWAYS
1810 *-----
0887- BC BC BC
088A- C5 CD D0
088D- D4 D9 A0
0890- D6 CF CC
0893- D5 CD C5
0896- BE BE BE 1820 MT .AS -/(<<<EMPTY VOLUME>>>)/
0899- 00 1830 .HS 00
1840 *-----

```

Don Lancaster's AWIIe TOOLKIT

Solve all of your Applewriter™ IIe hassles with these eight diskette sides crammed full of most-needed goodies including . . .

- Patches for NULL, shortline, IIc detrashing, full expansion
- Invisible and automatic microjustify and proportional space
- Complete, thorough, and fully commented disassembly script
- Detailed source code capturing instructions for custom mods
- Clear and useful answers to hundreds of most-asked questions
- Camera ready print quality secrets (like this ad, ferinstance)
- New and mind-blowing WPL routines you simply won't believe
- Self-Prompting (!) glossaries for Diablo, Epson, many others
- Includes a free "must have" bonus book and helpline service

All this and bunches more for only \$39.95. Everything is unlocked and unprotected. Order from SYNERGETICS, 746 First Street, Box 809-AAL, Thatcher, AZ, 85552. (602) 428-4073. VISA or MC accepted.

David Johnson challenged me a few days ago. We were talking about ProDOS: the need for a ProDOS version of the S-C Macro Assembler, the merits vs. enhanced DOS 3.3, and the rash of recent articles on shrinking various routines inside DOS to make room for more features.

I've been avoiding ProDOS as much as possible, trying not to notice its ever-increasing market-share. Dave's comment, "ProDOS is a fertile field for your shrinking talent," may have finally pushed me into action.

I am trying to make the ProDOS version of the S-C Macro Assembler, but is hard. I have Apple's manuals, Beneath Apple ProDOS, and the supplement to the latter book which explains almost every line of ProDOS code. Nevertheless, version 1.1.1 of ProDOS doesn't seem to conform to all these descriptions in every particular. I spent four hours last night chasing one little discrepancy. (Turned out to be my own bug, though.)

In the process, I ran across the subroutine ProDOS uses to convert binary numbers to decimal for printing. In version 1.1.1 it starts at \$A62F, and with comments looks like this.

```

1000 *SAVE S.PRODOS NUMOUT
1010 *-----
1020      .OR $A62F
1030      .TA $800
1040 *-----
1050 *   CONVERT 00.XX.AA FROM BINARY TO DECIMAL
1060 *   STORE UNITS DIGIT AT $201.Y
1070 *   STORE OTHER DIGITS AT SUCCESSIVE LOWER ADDRESSES
1080 *
1090 *       Note:  it is assumed and required that
1100 *              ACCUM+2 already by zeroed!
1110 *              Either that, or already set to the
1120 *              highest byte of a 24-bit value.
1130 *-----
1140 CONVERT.T0-DECIMAL
A62F- 8E B0 BC 1150      STX ACCUM+1
A632- 8D AF BC 1160      STA ACCUM
A635- 20 4D A6 1170      JSR DIVIDE.ACCUM.BY.TEN
A638- AD B2 BC 1180      LDA REMAINDER
A63B- 09 B0 1190      ORA #0
A63D- 99 01 02 1200      STA BUFFER+1,Y
A640- 88 1210      DEY
A641- AD AF BC 1220      LDA ACCUM      CHECK IF QUOTIENT ZERO
A644- 0D B0 BC 1230      ORA ACCUM+1
A647- 0D B1 BC 1240      ORA ACCUM+2
A64A- D0 E9 1250      BNE .1
A64C- 60 1260      RTS
1270 *-----
1280 DIVIDE.ACCUM.BY.TEN
A64D- A2 18 1290      LDX #24      24 BITS IN DIVIDEND
A64F- A9 00 1300      LDA #0      START WITH REM=0
A651- 8D B2 BC 1310      STA REMAINDER
A654- 20 D7 AA 1320      JSR SHIFT.ACCUM.LEFT
A657- 2E B2 BC 1330      ROL REMAINDER
A65A- 38 1340      SEC      REDUCE REMAINDER MOD 10
A65B- AD B2 BC 1350      LDA REMAINDER
A65E- E9 0A 1360      SBC #10
A660- 90 06 1370      BCC .2      STILL < 10
A662- 8D B2 BC 1380      STA REMAINDER
A665- EE AF BC 1390      INC ACCUM      QUOTIENT BIT
A668- CA 1400      DEX      NEXT BIT
A669- D0 E9 1410      BNE .1
A66B- 60 1420      RTS
1430 *-----

```

```

BCAF-      1440 ACCUM      .EQ $BCAF,BCB0,BCB1
BCB2-      1450 REMAINDER .EQ $BCB2
0200-      1460 BUFFER    .EQ $0200
           1470 *-----
           1480           .OR $AAD7
           1490           .TA $900
           1500 *-----
           1510 SHIFT.ACCUM.LEFT
AAD7- 0E AF BC 1520 ASL ACCUM
AADA- 2E B0 BC 1530 ROL ACCUM+1
AADD- 2E B1 BC 1540 ROL ACCUM+2
AAEO- 60      1550 RTS

```

The conversion routine is designed to handle values between 0 and \$FFFFFF. The highest byte must already have been stored at ACCUM+2 before calling CONVERT.TO.DECIMAL. The middle byte must be in the X-register, and the low byte in the A-register. The decimal digits will be stored in ASCII in the \$200 buffer, starting and \$201+Y and working backwards.

One way of converting from binary to decimal is to perform a series of divide-by-ten operations. After each division, the remainder will be the next digit of the decimal value, working from right to left. That is the technique ProDOS uses, and the division is done by the subroutine in lines 1280-1420.

The dividend is in ACCUM, a 3-byte variable. The low byte is first, then the middle, and finally the high byte. One more byte is set aside for the remainder. A 24-step loop is set up to process all 24 bits of ACCUM. In the loop ACCUM and REMAINDER are shifted left. If REMAINDER is 10 or more, it is reduced by ten and the next quotient bit set to 1; otherwise the next quotient bit is 0.

The first possible improvement I noted was in the area of lines 1330-1360. the ROL REMAINDER will always leave carry status clear, because we never let REMAINDER get larger than 9. If we delete the SEC instruction, and change SBC #10 to SBC #9 (because carry clear means we need to borrow), we can save one byte. But that's not really worth the effort.

Next I realized that REMAINDER could be carried in the A-register within the 24-step loop, and not stored until the end of the loop. Here is that version, which saves seven bytes (original = 31 bytes, this one = 24 bytes):

```

           1260 DIVIDE.ACCUM.BY.TEN
081E- A2 18 1270 LDY #24      24 BITS IN DIVIDEND
0820- A9 00 1280 LDA #0      START WITH REM=0
0822- 20 3A 08 1290 .1 JSR SHIFT.ACCUM.LEFT
0825- 2A      1300 ROL
0826- C9 0A 1310 CMP #10
0828- 90 05 1320 BCC .2      STILL < 10
082A- E9 0A 1330 SBC #10
082C- EE 36 08 1340 INC ACCUM      QUOTIENT BIT
082F- CA 1350 .2 DEX      NEXT BIT
0830- D0 F0 1360 BNE .1
0832- 8D 39 08 1370 STA REMAINDER
0835- 60      1380 RTS

```

To make sure my version really worked, I re-assembled the conversion program with an origin of \$800, and appended a little test program. Here is my test program, which converts the value at \$0000...0002 and prints it out.

```

0844- A5 00      1510 T      LDA 0
0846- 8D 38 08   1520      STA ACCUM+2
0849- A6 01      1530      LDX 1
084B- A5 02      1540      LDA 2
084D- A0 0A      1550      LDY #10
084F- 20 00 08   1560      JSR CONVERT.TO.DECIMAL
0852- C8         1570 .1    INY
0853- B9 01 02   1580      LDA BUFFER+1,Y
0856- 20 ED FD   1590      JSR $FDED
0859- C0 0A      1600      CPY #10
085B- 90 F5      1610      BCC .1
085D- 60         1620      RTS

```

My best version is yet to come. I considered the fact that we could SHIFT the next quotient bit into the low end of ACCUM rather than using INC ACCUM to set a one-bit. I rearranged the loop so that the remainder reduction was done first, followed by the shift-left operation. I had to change the remainder reduction to work modulo 5 rather than 10, because the shifting operation came afterwards. I also had to include my own three lines of code to ROL ACCUM, since the little subroutine in ProDOS started with ASL ACCUM. The result is still shorter than 31 bytes, but only four bytes shorter. Nevertheless, it is faster and neater, in my opinion.

```

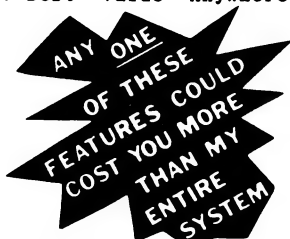
085E- A2 18      1640 DIVIDE.ACCUM.BY.TEN.SHORTEST
0860- A9 00      1650      LDX #24      24 BITS IN DIVIDEND
0862- C9 05      1660      LDA #0        START WITH REM=0
0864- 90 02      1670 .1    CMP #5
0866- B9 05      1680      BCC .2        STILL < 10
0868- 2E 36 08   1690      SBC #5
086B- 2E 37 08   1700 .2    ROL ACCUM
086E- 2E 38 08   1710      ROL ACCUM+1
0871- 2A         1720      ROL ACCUM+2
0872- CA         1730      ROL
0873- D0 ED      1740      DEX          NEXT BIT
0875- 8D 39 08   1750      BNE .1
0878- 60         1760      STA REMAINDER
0879- 60         1770      RTS

```

BLANKENSHIP BASIC

For the Apple II+, IIe, and IIC (3.3 DOS)

1. Full Interpreter, not a pre-processor
2. WHILE-ENDWHILE and REPEAT-UNTIL loops
3. True IF-THEN-ELSE-ENDIF, (Using WHEN)
4. PRINT.USING, FILE, MERGE, RANDOMIZE
5. PRINT and TAB commands work in HIRES
6. 80 columns supported on IIe and IIC
7. Full Editor with AUTO-NUM and RENUM
8. Listings are indented automatically
9. Fast SORT, SEARCH and INSTR\$ commands
10. BOX, BOXFILL, DRAW.USING, and SOUND
11. No more CHR\$(4) for DOS commands
12. DEFINE and PERFORM named procedures
13. 99% upward compatible with Applesoft
14. Best Value Anywhere - Guaranteed!



```

1000 REM sample listing
1010 COMPILE
1020 PERFORM "INPUT DATA"
1030 REPEAT
1040   PERFORM "DATA CHECK"
1050   WHEN A > 100 THEN
1060     PRINT "BIG NUMBER"
1070   ELSE
1080     PRINT "SMALL NUMBER"
1090     A=A+1
1100   ENDWHEN
1110 UNTIL A > 200
1120 END
1130 DEFINE "INPUT DATA"
1140   REM this is a dummy
1150   REM   procedure
1160 FINISH
1170 DEFINE "DATA CHECK"
1180   REM so is this
1190 FINISH

```

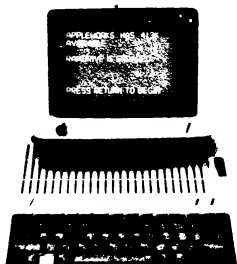
Money back if not entirely satisfied!

mail check to:
John Blankenship
P.O. Box 47934
Atlanta GA 30362

\$25

postpaid

The Apple IIc with Z-RAM



Applied Engineering and Apple computer have teamed up to take your IIc to new heights.

Applied Engineering's Z-RAM card for the IIc is available with 256K or 512K of additional memory and a powerful Z-80 microprocessor for running CP/M software.

Z-RAM fits neatly inside the IIc. Installation is easy, clear instructions show you how. You'll need a screwdriver and about 10 minutes (if you can change a light bulb you can install Z-RAM).

Z-RAM and Appleworks will knock your socks off.

A 256K Z-RAM will give you a 229K available desktop and Appleworks will be completely loaded into memory. Appleworks will now run about 10 times faster in your IIc with 1 disk drive than in other IIc's with two disk drives. A 512K Z-RAM will give you a 413K available desktop. A 256K Z-RAM can be upgraded to 512K at a later date when your needs grow.

Z-RAM is also a high speed solid state disk drive. With Z-RAM your programs will load and save over 20 times faster. Z-RAM's RAM disk is compatible with Applesoft, Pro-DOS, DOS 3.3, PASCAL and CP/M. And with Z-RAM you can copy a disk in one pass. Just insert the original, remove the original, insert blank disk! That's it! Z-RAM is another disk drive, only 20 times faster, 4 times larger capacity, and no whirring, clicking or waiting!

But before you start panting over all that extra memory, don't forget that the Z-RAM card has a built-in high speed Z-80 processor chip that allows you to run CP/M programs like Wordstar, dBASE II, Turbo PASCAL, Microsoft BASIC, FORTRAN and COBOL and over 3,000 other CP/M programs. So Z-RAM not only makes Apple programs run better and faster, it lets you run MORE programs.

With the Z-RAM card installed, your IIc is still your IIc only now you'll have that extra memory that Appleworks and other programs need. And you can run all that great CP/M software that others can only dream about.

Z-RAM is 100% compatible with all IIc software and hardware including the mouse, 2nd disk, modem and printer. Z-RAM is easily handled by the IIc power supply as power consumption is kept very low by using two custom integrated circuits and a patent pending power saving design. And Z-RAM is from Applied Engineering, the acknowledged leader and innovator of accessories for the Apple.

Z-RAM comes complete with manual, RAM disk software, Z-80 operating system, CP/M manual and a 3 year no hassle warranty.

So the next time somebody asks you why you didn't get an IBM P.C. tell him you bought a IIc because the IBM didn't have enough memory and was too slow and couldn't run CP/M software. And tell him you made it past the 8th grade.

Z-RAM with 256K **\$449**

Z-RAM with 512K **\$649**

Call (214) 492-2027

9 a.m. to 11 p.m. 7 days a week

Send check or money order to:

Applied Engineering

P. O. Box 798

Carrollton, Texas 75006



MasterCard.



Visa and

C.O.D. welcome. No extra charge for credit cards. Texas residents add 5 1/2% sales tax. Add \$10.00 if outside U.S.A.



APPLIED ENGINEERING

Fast Text Windows for Applesoft.....Michael Ching
2118 Kula Street, Honolulu, HI 96817

The program WINDER by Mike Seeds in the January 1985 NIBBLE was found to be very interesting. This was especially so because we, coincidentally, had been working on a similar routine for use in an upcoming strategy sports game.

The main difference between our programs was that the routines used in WINDER are written completely in Applesoft, and thus suffer from the relatively slow speed of the Applesoft interpreter. This is especially evident in the opening of the windows. Our routine, on the other hand, is written in assembly language and executes more quickly.

There are a couple of other major differences. Seeds' routine saves the text, to be overwritten by the window, in a string array WS\$. Our routine saves the text in the secondary text page (memory locations \$800 through \$BFF). One advantage of doing this is that more than one window can be opened at the same time (although the windows may not overlap). A disadvantage is that the secondary text page occupies the same space that an Applesoft program normally would start at. This makes it necessary to relocate the Applesoft program above the secondary text page.

Another difference is that WINDER specifies the window dimensions with the width and height of the window, along with the top and left coordinates. We chose to specify directly the top, bottom, left, and right boundaries.

The assembly language routine is called by the familiar & followed by the appropriate parameters. The format is & WT,WB,WL,WR,TP where WT is the top coordinate of the window, WB is the bottom coordinate, WL is the left coordinate, WR is the right coordinate, and TP is the text page number. If TP is set to 1, the text to be replaced by the window is saved to the secondary text page and the window is formed. If TP is set to 2, the text is restored to the primary text page from the secondary text page. At present, there is no error checking of the parameter values, and care must be taken to ensure that WB is set greater than WT, and WR greater than WL.

The program is assembled to load into the tail end of the input buffer and the free space in page 3 (\$2F5-3C9). The portion inside page 2 is only used to set up the ampersand hook, so it is not a problem if this code gets wiped out by long input lines after loading. This setup is done in lines 1250-1290.

Lines 1320-1470 perform the task of getting the parameter values from Applesoft and placing them into temporary storage. The routines GETBYT and COMBYTE are used, and will evaluate expressions used in the calling Applesoft program. The width of the window is also calculated here. The text page value is decremented by one for ease of future manipulation. Line 1340 initializes the beginning of a loop which will copy the characters in the designated text page to the opposite text page,

Lines 1500-1510 call the monitor routine BASCALC. BASCALC calculates the starting (leftmost) memory address of the screenline, and stores it in the pointers BASL and BASH.

Lines 1520-1640 set up two pointers, one in the real screen and one in the alternate screen area. The pointers point to the beginning of the current line starting at the left edge of the caller's window. A1 points at the source, and A2 at the destination, for a move loop which will copy the characters within the window on the current line.

The destination address is the source address offset by \$400 (up or down depending on the source text page). The calculation is done by exclusive ORing the source address with #\$0C (or 00001100 in binary). For example, if BASH was \$07, exclusive ORing will yield \$0B. If it was \$0B, exclusive ORing will yield \$07.

Lines 1660-1700 comprise the move loop.

Lines 1720-1850 check to see if the frame of the window needs to be drawn. If the text page is being restored (window being closed), then the frame routine is skipped. If the window is being cleared, the frame is drawn.

First I store an inverse blank at each end of the line, which is sufficient for all except the top and bottom lines. Then I check: if it is the top or bottom line, I fill in the rest of the line with inverse blanks.

Lines 1870-1900 check whether the entire window has been processed. If not, the program loops back to process the next line.

Lines 1920-2050 check to see whether the window boundaries need to be set. If the window is being opened (TPAGE = 0), then they are set, and HOME clears out the window. Note that the window parameters are set so that the frame is outside it.

```

1000 *SAVE S.WINDOWS
1010 *-----
1020 * MOVE WINDOW
1030 * by Mike Ching, Kula Software
1040 * 2118 Kula Street. Honolulu, HI 96817
1050 *-----
20- 1060 WNDLFT .EQ $20
21- 1070 WNDWIDTH .EQ $21
22- 1080 WNDTOP .EQ $22
23- 1090 WNDBTM .EQ $23
28- 1100 BASL .EQ $28
29- 1110 BASH .EQ $29
18- 1120 A1 .EQ $18,19 MEMORY SOURCE START
1A- 1130 A2 .EQ $1A,1B MEMORY SOURCE END
1140 *-----
03F5- 1150 AMPERV .EQ $3F5
1160 *-----
E6F8- 1170 GETBYT .EQ $E6F8
E74C- 1180 COMBYTE .EQ $E74C
FBC1- 1190 BASCALC .EQ $FBC1
FC58- 1200 HOME .EQ $FC58

```

```

1210 *-----
1220 .OR $2F5
1230 .TF B.WINDOWS
1240 *-----
02F5- A9 00 1250 SETUP LDA #MOVE.WINDOW SET UP & VECTOR
02F7- 8D F6 03 1260 STA AMPERV+1
02FA- A9 03 1270 LDA /MOVE.WINDOW
02FC- 8D F7 03 1280 STA AMPERV+2
02FF- 60 1290 RTS
1300 *-----
1310 MOVE.WINDOW
0300- 20 F8 E6 1320 JSR GETBYT GET VALUES FROM APPLESOFT
0303- 8E 9F 03 1330 STX TOP
0306- 8E A4 03 1340 STX LINE
0309- 20 4C E7 1350 JSR COMBYTE
030C- 8E A0 03 1360 STX BOTTOM
030F- 20 4C E7 1370 JSR COMBYTE
0312- 8E A1 03 1380 STX LEFT
0315- 20 4C E7 1390 JSR COMBYTE
0318- 8E A2 03 1400 STX RIGHT
031B- 38 1410 SEC WIDTH = RIGHT-LEFT
031C- 8A 1420 TXA
031D- ED A1 03 1430 SBC LEFT
0320- 8D A3 03 1440 STA WIDTH
0323- 20 4C E7 1450 JSR COMBYTE GET DIRECTION (1 OR 2)
0326- CA 1460 DEX
0327- 8E A5 03 1470 STX TPAGE
1480 *-----
1490 MOVE.LINE
032A- AD A4 03 1500 LDA LINE BASL,H = BASCALC(LINE)
032D- 20 C1 FB 1510 JSR BASCALC
0330- 18 1520 CLC
0331- A5 29 1530 LDA BASH
0333- AE A5 03 1540 LDX TPAGE
0336- F0 02 1550 BEQ .1 ...SOURCE IS REAL SCREEN
0338- 49 0C 1560 EOR #$0C ...SOURCE IS SAVED SCREEN
033A- 85 19 1570 STA A1+1 SOURCE HI BYTE
033C- 49 0C 1580 EOR #$0C FLIP TEXT PAGE
033E- 85 1B 1590 STA A2+1 DESTINATION HI BYTE
0340- 18 1600 CLC MEMSTART = BASL,H + LEFT
0341- A5 28 1610 LDA BASL
0343- 6D A1 03 1620 ADC LEFT
0346- 85 18 1630 STA A1 SOURCE LO BYTE
0348- 85 1A 1640 STA A2 DESTINATION LO BYTE
1650 *---MOVE THE LINE SEGMENT-----
034A- AC A3 03 1660 LDY WIDTH
034D- B1 18 1670 LDA (A1),Y
034F- 91 1A 1680 STA (A2),Y
0351- 88 1690 DEY
0352- 10 F9 1700 BPL .2
1710 *---IF CLEARING, DRAW FRAME-----
0354- AC A5 03 1720 LDY TPAGE
0357- D0 1B 1730 BNE .4 ...NOT CLEAR. DO NOT DRAW FRAME
0359- A9 20 1740 LDA #$20 INVERSE BLANK
035B- 91 18 1750 STA (A1),Y LEFT SIDE
035D- AC A3 03 1760 LDY WIDTH
0360- 91 18 1770 STA (A1),Y RIGHT SIDE
0362- AE A4 03 1780 LDX LINE
0365- EC 9F 03 1790 CPX TOP
0368- F0 05 1800 BEQ .3 ...TOP LINE
036A- EC A0 03 1810 CPX BOTTOM
036D- D0 05 1820 BNE .4 ...NEITHER TOP NOR BOTTOM
036F- 91 18 1830 STA (A1),Y
0371- 88 1840 DEY
0372- D0 FB 1850 BNE .3
1860 *---NEXT LINE-----
0374- EE A4 03 1870 INC LINE UNTIL LINE > BOTTOM
0377- AD A0 03 1880 LDA BOTTOM
037A- CD A4 03 1890 CMP LINE
037D- B0 AB 1900 BCS MOVE.LINE ANOTHER LINE TO MOVE
1910 *---IF CLEARING, SET WINDOW-----
037F- AD A5 03 1920 LDA TPAGE
0382- D0 1A 1930 BNE .5
0384- AE A1 03 1940 LDX LEFT
0387- E8 1950 INX
0388- 86 20 1960 STX WNDLFT

```

MORE RAMWORKS™ MEMORY!

Everything needed, including easy instructions on expanding your Ramworks Card with MORE MEMORY FOR YOUR APPLE IIe! All chips are highest quality & guaranteed three years. Can be used with other memory cards and printer buffers also - CALL FOR DETAILS.

	List Price	Our Price
64k Ramworks expander kit	70.	47.
128k Ramworks expander kit	140.	92.
256k Ramworks expander kit	250.	162.
512k Ramworks expander kit	500.	314.
1024k Ramworks expander kit	1099.	659.
64k Ramworks Card (complete)*	179.	158.
128k Ramworks Card (complete)*	249.	199.
256k Ramworks Card (complete)*	399.	299.
512k Ramworks Card (complete)*	649.	485.
1024k Ramworks Card (complete)*	1199.	899.

*** Includes FREE AppleWorks Expander software!**

Appleworks Program	250.	210.
Imagewriter (wide carr/acc kit)	749.	626.

Terms: For fastest delivery send Cashier's or Certified check, or Money Order. C.O.D. and personal checks accepted (allow 14 days to clear). Add 3% for MasterCard/Visa (include # and expiration). Add \$3 shipping. Texas residents add 6½% sales tax.

Ramworks, AppleWorks/Imagewriter trademarks of Applied Engineering & Apple Computer respectively. Dealer inquiries welcome.

COIT VALLEY COMPUTERS

14055 Waterfall Way

(214) 234-5047

Dallas, Texas 75240

```

038A- AE A3 03 1970      LDX WIDTH
038D- CA      1980      DEX
038E- 86 21      1990      STX WNDWDTH
0390- AE 9F 03 2000      LDX TOP
0393- E8      2010      INX
0394- 86 22      2020      STX WNDTOP
0396- AE A0 03 2030      LDX BOTTOM
0399- 86 23      2040      STX WNDBTM
039B- 20 58 FC 2050      JSR HOME
039E- 60      2060      RTS
          2070      #-----
039F-      2080 TOP      .BS 1      PROGRAM STORAGE
03A0-      2090 BOTTOM .BS 1
03A1-      2100 LEFT  .BS 1
03A2-      2110 RIGHT .BS 1
03A3-      2120 WIDTH .BS 1
03A4-      2130 LINE  .BS 1
03A5-      2140 TPAGE .BS 1
          2150      #-----

```

The next listing shows the revised WINDER routine using the assembly language routines. Line 40 checks to see if the program has been relocated above the secondary text page. If not, the start of program pointers are changed and the program is re-RUN. This causes DOS to position the program above the secondary text page. Line 50 BRUNS the assembly language routine.

The program is really quite different from that of Mike Seeds, as you can see if you compare them. Clearing and restoring windows is now very efficient, due to the &-routine. I moved the delay and closing logic into a common subroutine. I also added a randomly sized and positioned window in lines 400-410.

```

10 REM WINDOW DEMO PROGRAM, BASED ON PROGRAM
20 REM BY MIKE SEEDS NIBBLE. JAN 1985
30 REM -----
40 P = 12: IF PEEK (104) < P THEN POKE 104,P: POKE P * 256.0: PRINT CHR$
(4)"RUN WINDOW DEMO"
50 PRINT CHR$(4)"BRUN B.WINDOWS"
60 REM -----
100 TEXT : HOME
110 FOR I = 1024 TO 2047 STEP 128: FOR J = 0 TO 119: POKE I + J, RND (1)
* 26 + 193: NEXT J,I
120 PRINT " WINDOW DEMONSTRATION";: CALL - 868: PRINT : CALL -
868
130 VTAB 22: CALL - 958: PRINT : PRINT " PRESS ANY KEY TO HALT";
140 REM -----
150 T = 10:B = 14:L = 12:R = 21: & T,B,L,R,1: REM OPEN WINDOW
160 PRINT " TINY WINDOW": GOSUB 1000: REM DELAY AND CLOSE WINDOW
170 REM -----
200 T = 2:B = 7:L = 6:R = 31: & T,B,L,R,1
210 VTAB T + 3: HTAB 4: PRINT "NOTICE THE TEXT IS": HTAB 4: PRINT "RESTO
RED CORRECTLY."
220 GOSUB 1000
230 REM -----
260 T = 10:B = 19: & T,B,L,R,1
270 FOR J = 1 TO 25: PRINT " ";J,J * J: NEXT J
280 PRINT : PRINT " SCROLLING IS AUTOMATIC"
290 GOSUB 1000
330 REM -----
400 W = RND (1) * 20 + 5:H = RND (1) * 10 + 5:T = RND (1) * (24 - H):B
= T + H:L = RND (1) * (40 - W):R = L + W
410 & T,B,L,R,1: PRINT "ABCDEFGHIJKLMNOPQRSTUVWXYZ": GOSUB 1000
420 GOTO 150
1000 FOR D = 1 TO 1500: NEXT
1010 & T,B,L,R,2: REM CLOSE WINDOW
1020 IF PEEK (- 16384) < 128 THEN RETURN
1030 POP : POKE - 16368,0: TEXT : HOME : END

```

An 8086/8088 Cross Assembler.....Don Rindsberg

As one of S-C's avid fans, I have developed an 8086/8088 Cross Assembler for your Apple which will enable you to generate code to run on the IBM PC's and their clones as well as many other 16-bit machines. All the 8086/8088 instructions are covered as well as the multiplicity of addressing modes. The mnemonics are based on Microsoft's assembler. This assembler is based on S-C Assembler II Version 4.0 (the one before Macro Assembler), so it doesn't include the newer features like macros or the EDIT command. Documentation covering the differences from the 6502 version is included.

With Bob's permission, XSM 8086/8088 is available to owners of the S-C 6502 assembler (Version 4.0 or later) for \$80.00 post-paid. Included on the disk are sample source programs so you can become familiar with the syntax. Send personal check or money order (no credit cards or purchase orders) to:

The Bit Stop
5958 S. Shenandoah Rd.
Mobile, AL 36608
Attn: Don Rindsberg
(205) 342-1653

WINDOWS!

for the APPLE II computer

Windows is:

- Macintosh-like windows for the Apple II!
- Easy to use from Basic!
- Ten new commands!
- A special effects creator!
- A screen splitter!
- Compatible with all Apple II computers with 48k or more!
- Compatible with ProDOS and DOS 3.3
- Only \$19.95!

Display any message, menu, etc. right over the existing text without destroying the original display. Windows are completely independent. Send your check to:

|Jan Eugenides, 11601 N.W. 18th St.
Pembroke Pines, FL 33026

ASSEMBLY CORNER

assembly language for the beginner

~~~~~

Assembly Corner is now offering two accelerated courses on 6502 assembly language for those who are serious about learning to program.

- 1 Assembly for the serious beginner  
-four lessons each month on disk, along with programs, examples, source code, etc.  
Price: \$15/month
- 2 Graphics and Animation series  
-four quarterly disks full of information on Apple graphics, with special utilities not available elsewhere

Price \$15/disk

Assembly language runs 50 TIMES faster than Basic! Unleash the power of your Apple. Learn to program in it's native tongue.

-----  
| Yes, I want to learn! Send me! |  
| \_ The beginners' series (one month) \$15 |  
| \_ The beginners' series (one year) \$150 |  
| \_ The Graphics series (one disk) \$15 |  
| \_ The Graphics series (one year) \$50 |  
| A free assembly language fact poster |  
| is included with each order!! |  
|-----

### ASSEMBLY CORNER

11601 N.W. 18th St.  
Pembroke Pines, FL 33026  
-----

A Powerful 65816 Board on the Horizon.....Bob Sander-Cederlof

Some of you may have heard of Micro Magic, a company in Maryland that is planning to produce a plug-in card for your Apple with fast RAM and a fast 65816. Well, if not, now you have.

I spoke yesterday with Will Troxell, and got an overview of their plans. He and Frank Krol are working together on the project. Their goal is to produce the most powerful and flexible card they can and yet still bring it in for a low price. The card will basically be similar to the Accelerator //e, in that it consists of a fast microprocessor, fast RAM, and the logic to take control away from the 6502 or 65C02 on your Apple motherboard.

But instead of a 65C02 running at 3.58 MHz, you will get a 65816 running at 6 MHz. Instead of one row of RAM chips, you get two. Troxell's board will probably come with 64K or 128K of 6MHz dynamic RAM, but later this year they have been promised that 256K RAMs fast enough for 6 MHz operation will be in production; then you will be able to expand your board to 256K or 512K bytes of RAM.

There is a firmware socket on the board which can accept a 27128 (16K bytes of firmware, the same as you find in a //c). They do not plan to include any firmware at the beginning, but it certainly can be filled up with your own goodies.

There are two external connectors on the board. One of these allows you to add another 512K RAM. Remember, this is directly addressable RAM, not bank-switched. The 65816 can directly address up to 16 megabytes, with its 24-bit address bus.

It is also exciting to remember that a plain ol' 6502 running at 1 MHz (what you have now) is roughly equivalent in speed to most of the 8088 and Z-80 computers on the market. A 6 MHz 6502 could beat a 20MHz Z-80 (were they to make one so fast). A 6 MHz 65816 will beat out 68000's, 80286's, and so on. Why is this true? Because all those other chips use micro-programmed instruction sets, taking many clock cycles for each instruction. The 6502 and its progeny are fully implemented in hardware gates, so only a handful of clock cycles are needed.

Furthermore, a 65816 instruction will take from one to four bytes of memory, while a 68000 instruction will take 2, 4, 6, 8, or 10 bytes. Now I am not trying to deny the power of some of those 68000 instructions. One of them may take many steps in 65816 code. Especially if you need to deal with 32-bit operands. But it is my experience that those super instructions are relatively infrequent in practical programs. Most programs spend most of their time just moving bytes from here to there and back again.

Now if we could only get one! For about fifteen months we have been hearing "in two to four weeks". We could despair, were it not for our historical perspective. The same thing happened with the 65C02, and now we really do have them in abundance. By this time next year, you may be hearing solid confirmation

of the rumor (heard this week) that Apple and GTE are discussing large orders of 65816s.

But I digress. Back to Troxell and Krol. Their new board will be called the MAX-816, and a new operating system they are designing for it will be MAX-OS. A special circuit on the card will optimize memory re-mapping for both DOS and ProDOS, automatically, so that maximum possible use is made of the fast RAM on the card. The fewer times the card has to slow down to use motherboard RAM, the faster your programs fly.

MAX-OS will not be necessary for you to get a bang out of MAX-816, because it will work like the Accelerator //e and make most existing programs six times faster (exclusive of I/O). But when it is ready, it will open up new vistas, with RAM stretching out in every direction as far as the eye can see. In a design reminiscent of one from a certain large phone company, the kernel is written in assembly language, with a C-shell wrapped around it.

Personally, I am no great fan of complex operating systems. The simpler and smaller the better, in my book. I still like DOS 3.3, especially with enhancements I regularly patch in. Nevertheless it does take more management when you have the magnitude and variety of resources that will be in the Apple of the future. Maybe MAX-OS will be the winner.

If Will and Frank are whetting your appetite, you can write to them at Micro Magic, Box 281, Millersville, MD 21108. Or you might be able to reach them at (301) 987-6083.

## Now you can monitor and control the world (or at least your part of it) with a little help from **APPLIED ENGINEERING**

### 12 BIT, 16 CHANNEL, PROGRAMMABLE GAIN A/D

- All new 1984 design incorporates the latest in state-of-art I.C. technologies.
- Complete 12 bit A/D converter, with an accuracy of 0.02%!
- 16 single ended channels (single ended means that your signals are measured against the Apple's GND) or 8 differential channels. Most all the signals you will measure are single ended.
- 9 software programmable full scale ranges, any of the 16 channels can have any range at any time. Under program control, you can select any of the following ranges:  $\pm 10$  volts,  $\pm 5$  V,  $\pm 2.5$  V,  $\pm 1.0$  V,  $\pm 500$  mV,  $\pm 250$  mV,  $\pm 100$  mV,  $\pm 50$  mV, or  $\pm 25$  mV.
- Very fast conversion (25 micro seconds).
- Analog input resistance greater than 1,000,000 ohms.
- Laser-trimmed scaling resistors.
- Low power consumption through the use of CMOS devices.
- The user connector has +12 and -12 volts on it so you can power your sensors.
- Only elementary programming is required to use the A/D.
- The entire system is on one standard size plug in card that fits neatly inside the Apple.
- System includes sample programs on disk.

**PRICE \$319**

A few applications may include: the monitoring of ● flow ● temperature ● humidity ● wind speed ● wind direction ● light intensity ● pressure ● RPM ● soil moisture and many more.

### A/D & D/A

- Single PC card
- 8 channels A/D
- 8 channels D/A
- Superfast conversion time
- Very easy programming
- Many analog ranges
- Manual contains sample applications

#### A/D SPECIFICATIONS

- 0.3% accuracy
  - On-board memory
  - Fast conversion (0.78 MS per channel)
  - A/D process totally transparent to Apple (just poke the data)
  - User programmable input ranges are 0 to 10 volts, 0 to 5, -5 to +5, -2.5 to +2.5, -5 to 0, -10 to 0.
- The A/D process takes place on a continuous, channel sequencing basis. Data is automatically transferred to its proper location in the on-board RAM. No A/D converter could be easier to use.

#### D/A SPECIFICATIONS

- 0.1% accuracy
  - On-board memory
  - On-board output buffer amps can drive 5 MA
  - D/A process is totally transparent to the Apple (just poke the data)
  - Fast conversion (0.03 MS per channel)
  - User programmable output ranges are 0 to 5 volts and 0 to 10 volts
- The D/A section contains 8 digital to analog converters, with output buffer amplifiers and all interface logic on a single card. On-card latches are provided for each of the eight D/A converters. No D/A converter could be easier to use. The on-board amplifiers are laser-trimmed during manufacture, thereby eliminating any requirement for off-set nulling.

**PRICE \$199**

### SIGNAL CONDITIONER

Our 8 channel signal conditioner is designed for use with both our A/D converters. This board incorporates 8 F.E.T. op-amps, which allow almost any gain or offset. For example, an input signal that varies from 2.00 to 2.15 volts or a signal that varies from 0 to 50 mV can easily be converted to 0-10V output for the A/D.

The signal conditioner's outputs are on a high quality 16 pin gold I.C. socket that matches the one on the A/D's so a simple ribbon cable connects the two. The signal conditioner can be powered by your Apple or from an external supply.

#### FEATURES

- 4.5" square for standard card cage and 4 mounting holes for standard mounting. The signal conditioner does not plug into the Apple, it can be located up to 16 mile away from the A/D.
- 22 pin 156 spacing edge card input connector (extra connectors are easily available i.e. Radio Shack).
- Large bread board area.
- Full detailed schematic included.

**PRICE \$79**

### I/O 32

- Your inputs can be anything from high speed logic to simple switches
  - Programming is made very easy by powerful on-board firmware
  - The I/O 32 is your best choice for any control application
- The I/O manual includes many programs for inputs and outputs.

#### Some applications include:

- Burglar alarm, direction sensing, use with relays to turn on lights, sound buzzers, start motors, control tape recorders and printers, use with digital joystick.

**PRICE \$89**

Please see our other full page ad in this magazine for information on Applied Engineering's Timemaster Clock Card and other products for the Apple.

Our boards are far superior to most of the consumer electronics made today. All I.C.'s are in high quality sockets with mil-spec. components used throughout. P.C. boards are glass-epoxy with gold contacts. Made in America to be the best in the world. All products compatible with Apple II and //e.

Applied Engineering's products are fully tested with complete documentation and available for immediate delivery. All products are guaranteed with a no hassle three year warranty.

Texas Residents Add 5% Sales Tax  
Add \$10.00 if Outside U.S.A.

Send Check or Money Order to:  
**APPLIED ENGINEERING**  
P.O. Box 798  
Carrollton, TX 75006

Call (214) 492-3027  
7 a.m. to 11 p.m. 7 days a week  
MasterCard, Visa & C.O.D. Welcome  
No extra charge for credit cards

USR Command to List Major Labels Only.....Bob Sander-Cederlof

Sometimes when I am working with a large source file in the S-C Macro Assembler it would be nice to be able to list only those lines that define major labels. Seeing only them would give an overview of an entire file, and enable me to quickly find the section I want to work on.

A major label is one that starts with a letter. Local labels start with a period, macro private labels start with a colon. Lines might also start with an asterisk or semicolon, if they are comments, or with blank.

You can add commands to the Macro Assembler in several ways. One easy built in one is the USR command. A vector at \$D007 (or \$1007 with the low memory version) can point to the code to process a command of your own making. Lines 1080-1140 in the following listing set up the vector for my special USR command. Since it is in the high RAM area (sometimes called "language card"), I reference \$C083 twice to write enable the RAM.

Once the USR vector is loaded, typing a command "USR" will execute my code. When this happens, the entire command I typed will be in a buffer starting at \$200. Some routines exist inside S-C Macro which can help in parsing the command further and in implementing its functions, and I will use them in this example. If you have the source code to one of the S-C Macro versions, it is not too difficult to find these routines. And if you don't have it, you can always disassemble and analyze, a true form of adventure. The addresses shown in lines 1040-1060 correspond to version 2.0 of the S-C Macro Assembler.

Line 1165 calls on a subroutine I call PARSE.LINE.RANGE (PLR). PLR starts by setting up SRCP to point to the beginning of the source program, and ENDP to the end of same. Then it looks at the command line for various forms of line numbers. You might have none at all, in which case PLR is finished. You might have one number alone, or a period. (A period is shorthand for the last remembered line number.) That might be preceded by or followed by a comma. You might have two numbers separated by a comma. Here is a table showing what happens in each case:

|       | SRCP    | ENDP  | CARRY |
|-------|---------|-------|-------|
|       | ----    | ----  | ----- |
| none  | pstart  | pend  | set   |
| #     | #start  | #end  | clear |
| #,    | #start  | pend  | clear |
| ,#    | pstart  | #end  | clear |
| #1,#2 | #1start | #2end | clear |

where # means number or "."

pstart = address of start of source code

pend = address of end of source code

#start = address of starting line #

#end = address of ending line #

Line 1170 call a routine in the assembler to compare SRCP and ENDP to see if we are finished or not. The code is simply:





## FONT DOWNLOADER & EDITOR (\$39.00)

Turn your printer into a custom typesetter. Downloaded characters remain active while printer is powered. Use with any Word Processor program capable of sending ESC and control codes to printer. Switch back and forth easily between standard and custom fonts. All special printer functions (like expanded, compressed etc.) apply to custom fonts. Full HIRES screen editor lets you create your own characters and special graphics symbols. Compatible with many parallel printer I/F cards. User driver option provided. For Apple II, II+, //e. Specify printer: Apple Dot Matrix, C.Itoh 8510A (Prowriter), Epson FX 80/100, or OkiData 92/93.

**NEW !!!** The Font Downloader & Editor for the Apple Imagewriter Printer. For use with Apple II, II+, //e (with SuperSerial card) and the new Apple //c (with builtin serial interface).

**NEW !!! FONT LIBRARY DISKETTE #1 (\$19.00)** contains lots of user-contributed fonts for all printers supported by the Font Downloader & Editor. Specify printer with order.

## DISASM 2.2e - AN INTELLIGENT DISASSEMBLER (\$30.00)

Investigate the inner workings of machine language programs. DISASM converts machine code into meaningful, symbolic source. Creates a standard text file compatible with S-C, LISA, ToolKit and other assemblers. Handles data tables, displaced object code & even lets you substitute your own meaningful labels. (100 commonly used Monitor and Pg Zero names included.) An address-based triple cross reference table is provided to screen or printer. DISASM is an invaluable machine language learning aid to both novice & expert alike. Don Lancaster says DISASM is "absolutely essential" in his new ASSEMBLY COOKBOOK. For entire Apple II family including the new Apple //c (with all the new opcodes). **SOURCE CODE** available for an additional \$30.00

## S-C Assembler (Ver 4.0 only) SUPPORT UTILITY PACKAGE (\$30.00)

- \* SC.XREF - Generates a GLOBAL LABEL Cross Reference Table for complete documentation of source listings.
- \* SC.GSR - Global Search & Replace eliminates tedious manual renaming of labels. Search all/part of source.
- \* SC.TAB - Tabulates source files into neat, readable form. **SOURCE CODE** available for an additional \$30.00

## The 'PERFORMER' CARD (\$39.00)

Plugs into any slot to convert a 'dumb' centronics-type printer I/F card into a 'smart' one. Command menu eliminates need to remember complicated ESC codes. Features include perforation skip, auto page numbering with date & title. Includes large HIRES graphics & text screen dumps. Specify printer: MX-80 with Grafix-80, MX-100, MX-80/100 with Grafixplus, NEC 8092A, C.Itoh 8510 (Prowriter), OkiData 82A/83A with Okigraph & OkiData 92/93. **SOURCE CODE: \$30.00**

## FIRMWARE FOR APPLE-CAT: The 'MIRROR' ROM (\$25.00)

Communications ROM plugs directly into Novation's Apple-Cat Modem card. Basic modes: Dumb Terminal, Remote Console & Programmable Modem. Features include: selectable pulse or tone dialing, true dialtone detection, audible ring detect, ring-back, printer buffer, 80 col card & shift key mod support. Uses superset of Apple's Comm card and Micromodem II commands. **SOURCE CODE: \$50.00**

## RAM/ROM DEVELOPMENT BOARD (\$30.00)

Plugs into any Apple slot. Holds one user-supplied 2Kx8 memory chip (6116 type RAM for program development or 2716 EPROM to keep your favorite routines on-line). Maps into \$Cn00-CnFF and \$C800-CFFF.

## NEW !!! C-PRINT For The APPLE //c (\$99.00)

Connect standard parallel printers to an Apple //c. C-PRINT is a hardware accessory that plugs into the standard Apple //c printer serial port. The other end plugs into any printer having a standard 36 pin centronics-type parallel connector. Just plug in and print! High speed data transfer at 9600 Baud. No need to reconfigure serial port or load software drivers for text printing.

Avoid a \$3.00 postage/handling charge by enclosing full payment with order. (Mastercard & VISA excluded)

**RAK-WARE 41 Ralph Road W. Orange N J 07052 (201) 325-1885**



```

LDA SRCP
CMP ENDP
LDA SRCP+1
SBC ENDP+1

```

Lines 1200-1210 pick up the first character after the line number. The source line format in memory is one byte for a byte count, two bytes for the line number, the text of the line, and a final terminating 00 byte. The blank which follows just after the line number in listings is not actually stored.

Characters in a source line are stored in "low" ASCII, values between \$01 and \$7F. Values from \$81 through \$BF indicate 1 to 63 blanks. The value \$C0 indicates repetitions of some other character. The byte following a \$C0 is the repetition count, and the byte after that is the character to be repeated. Lines 1220-1240 check for blanks and repeat tokens. Lines 1340-1350 pick up the repeated character if we found a repeat token.

Lines 1360-1390 check if the first character is a letter. If not, this line will not be listed. Lines 1250-1320 are executed to skip over the current line without listing it. Since the first byte of the line has a byte count, it is added to SRCP to move up the next line.

At line 1400 I call LIST CURRENT LINE to ... you guessed it. This subroutine also advances SRCP, so after it is finished I jump back to the top to check pointers and get the next line.

After assembling the program, I type MGO INIT to hook it in. Then "USR 1070," would list just lines 1080 and 1160.

```

1000 *SAVE S.LIST MAJOR LABELS
1010 *-----
DD-      1020 SRCP .EQ $DD,DE
1030 *-----
DEAF-    1040 PARSE.LINE.RANGE .EQ $DEAF OR 1EAF
DF11-    1050 CMP.SRCP.ENDP .EQ $DF11 OR 1F11
D737-    1060 LIST.CURRENT.LINE .EQ $D737 OR 1737
1070 *---LINK COMMAND-----
0800- AD 83 C0 1080 INIT LDA $C083 ENABLE LANGUAGE CARD
0803- AD 83 C0 1090 LDA $C083
0806- A9 11 1100 LDA #USR.LIST SET UP USR VECTOR
0808- 8D 07 D0 1110 STA $D007
080B- A9 08 1120 LDA /USR.LIST
080D- 8D 08 D0 1130 STA $D008
0810- 60 1140 RTS
1150 *---USR COMES HERE-----
1160 USR.LIST
0811- 20 AF DE 1165 JSR PARSE.LINE.RANGE
0814- 20 11 DF 1170 .1 JSR CMP.SRCP.ENDP
0817- 90 01 1180 BCC .2
0819- 60 1190 RTS
081A- A0 03 1200 .2 LDY #3 POINT TO FIRST CHAR
081C- B1 DD 1210 LDA (SRCP),Y
081E- 10 17 1220 BPL .5 NOT TOKEN
0820- C9 C0 1230 CMP #$C0
0822- B0 0F 1240 BCS .4 REPEAT TOKEN
0824- A0 00 1250 .3 LDY #0 SKIP TO NEXT LINE
0826- B1 DD 1260 LDA (SRCP),Y LINE LENGTH
0828- 18 1270 CLC
0829- 65 DD 1280 ADC SRCP
082B- 85 DD 1290 STA SRCP
082D- 90 E5 1300 BCC .1
082F- E6 DE 1310 INC SRCP+1
0831- D0 E1 1320 BNE .1 ...ALWAYS
1330 *-----

```

```

0833- A0 05 1340 .4 LDY #5 POINT AT RPTD CHAR
0835- B1 DD 1350 LDA (SRCP),Y
0837- C9 41 1360 .5 CMP #'A'
0839- 90 E9 1370 BCC .3 NOT LETTER
083B- C9 5B 1380 CMP #'Z'+1
083D- B0 E5 1390 BCS .3 NOT LETTER
083F- 20 37 D7 1400 JSR LIST.CURRENT.LINE
0842- 4C 14 08 1410 JMP .1
1420 #-----

```

## Review of the FCP Hard Disk.....Bob Sander-Cederlof

First Class Peripherals has been advertising for some months now their 10 megabyte hard disk system (The Sider) for the Apple. At only \$695, including drive, controller, cable, and software, it sounds too good to be true. We called them and asked for a chance to write a review, and they loaned us one for a month.

I first tried hooking it up to an Apple II Plus, the same one we have used with hard disks in the past. However, after 5 or 6 wasted hours, it still would not function. We could not even get the disk to completely initialize. I finally called the 800 number for customer service, and found out that there have been problems hooking the Sider to some II+'s. They suggested trying it on a //e before giving up. Sure enough, it worked perfectly on our //e. The Sider is sold subject to a 15-day trial period, so there is plenty of time to find out if it will work with your II+.

I am very pleased. The Sider works well, looks good, and is not too noisy. We have heard of at least one customer who did complain of the noise level, but I have never listened to a quieter one. Because of the venting design there is no internal fan, so the only noise is the spinning disk. Anyway, my office already has two fans going on Apples and another in a Minolta copier. The Sider nicely masks them all.

The size and shape are nice, too. It is somewhat smaller than I expected: less than 4x8x16 inches. At first I set it along side of my Apple (after all it is called the Sider), but now it is along the back edge of my work table. This way it takes practically no space at all, yet I can still easily reach the on/off switch.

The installation software that comes with the Sider initializes the 10 megabytes into four separate partitions. One is for DOS, one for ProDOS, one for CP/M, and one for Pascal. You can vary the partition size for each one, although a certain minimum amount must be allocated; you cannot squeeze one all the way out. The DOS partition allows a combination of floppy size volumes and large volumes. The large volumes give you three times the amount of a regular Apple floppy. I set mine up with 32 small volumes and one large volume.

The ProDOS partition divides the allocated space into two equal size volumes, designated /HARD1/ and /HARD2/. Since I shrank CP/M and Pascal to the minimum, the ProDOS volumes are about 2.5 megabytes each.

If you want to change the partitions, you have to completely re-initialize. That means all your files will disappear. Of course you can restore them from your backup floppy copies.

The only modification to DOS 3.3 that the Sider makes is to put a call to their firmware at \$BD00. I decided to apply my own set of patches, which among other things speed up LOAD, BLOAD, RUN, and BRUN. They were not only compatible, they even speeded up the hard disk! Here is a table comparing the Sider with floppies, both with and without my patches:

| BLOAD<br># sectors | ----floppies----- |         | ----The Sider---- |         |
|--------------------|-------------------|---------|-------------------|---------|
|                    | standard          | patched | standard          | patched |
| 22                 | 7.7               | 3.8     | 3.0               | 1.3     |
| 69                 | 18.7              | 5.6     | 6.7               | 2.4     |
| 131                | 32.6              | 8.6     | 12.3              | 3.8     |

I also timed the assembly of a large program, whose source was on two disks (the S-C Macro Assembler itself, in fact). With my speed up patches the floppy assembly took 4 minutes 50 seconds; the Sider with standard DOS took 3 minutes 50 seconds; the Sider with my patches took only 2 minutes 32 seconds.

All these times are under DOS 3.3 of course. ProDOS is about the same as my patched version of DOS in speed, but has other advantages like larger volumes and files.

The main competition for the Sider comes from the two most popular companies, Apple and Corvus. Apple's ProFILE hard disk is sleek and nice, and only costs three times what the Sider does. Since you are paying more, you also get less: Apple only supports ProDOS. The ProFILE doesn't work with CP/M, Pascal, or DOS 3.3. (Unless there is a new ProDOS compatible Pascal.) Corvus costs even more than ProFILE, last time I checked. On the other hand, they have an excellent reputation.

Its always hard to trust some new little company, even when they have a great product and price. Just who is First Class Peripherals, anyway? Well, they are a subsidiary of Xebec, one of the bigger makers of hard disks. Xebec has been around a long time (over ten years) and has a first class reputation. I think we can depend on them. The Sider comes with a one-year limited warranty, which I think means that if it breaks you send it in and they will fix it or replace it. (Note: a whole year, not just 90 days!) After the warranty has expired there is a flat \$150 charge for repairs.

The only way to buy a Sider is directly from First Class Peripherals. You can call them at 1-800-538-1307, or write to 2158 Avenue C, Bethlehem, PA 18001. If you are in a user group of significant size, I understand someone at FCP might want to visit with a demo unit. You might give them a call.

Apple Assembly Line is published monthly by S-C SOFTWARE CORPORATION, P.O. Box 280300, Dallas, Texas 75228. Phone (214) 324-2050. Subscription rate is \$18 per year in the USA, sent Bulk Mail; add \$3 for First Class postage in USA, Canada, and Mexico; add \$14 postage for other countries. Back issues are available for \$1 80 each (other countries add \$1 per back issue for postage).

All material herein is copyrighted by S-C SOFTWARE CORPORATION, all rights reserved. (Apple is a registered trademark of Apple Computer, Inc.)